

HERIOT-WATT UNIVERSITY
Department of Computer Science

A PLAIN MANS
GUIDE
TO ICL 4130
ALGOL

A. BALFOUR
Revised January 1972

HERIOT-WATT UNIVERSITY, EDINBURGH

DEPARTMENT OF COMPUTER SCIENCE

A PLAIN MAN'S GUIDE

T O

ICL 4130 ALGOL

A. BALFOUR

REVISED JANUARY, 1972.

LIST OF CONTENTS

1.0	<u>INTRODUCTION</u>		1
2.0	<u>THE STRUCTURE OF AN ALGOL JOB DECK</u>		2
3.0	<u>THE ICL 4100 REPRESENTATION OF ALGOL 60 SYMBOLS</u>		4
4.0	<u>ICL 4100 ALGOL INPUT/OUTPUT STATEMENTS</u>		5
	4.1 The Input of Information .		6
	4.2 The Output of Results ..		9
	4.3 The Output of Text ..		11
	4.4 Setting Procedures ..		13
	4.4.1 Prefix Setting Procedures		14
	4.4.2 Format Setting Procedures		15
	4.4.3 Device Setting Procedures		18
	4.4.4 Parameters out of Range		19
	4.4.5 Alarm Printing .		20
5.0	<u>A SAMPLE JOB FOR THE DES BATCH OPERATING SYSTEM</u>		20
6.0	<u>RESTRICTIONS ON ALGOL 60</u>		22
7.0	<u>ADDITIONS TO ALGOL 60</u>		23
8.0	<u>RANGE AND ACCURACY OF NUMBERS</u>		24
9.0	<u>HINTS ON EFFICIENCY</u>		24

10.0	<u>THE ALGOL ERROR DIAGNOSIS SYSTEM</u>	..	..	..	..	..	..	..	27
	10.1	Compile Time Errors	..	..	..	..	..	..	27
	10.2	Run Time Errors	..	..	..	..	..	..	28
	10.3	State of a Job on Completion	..	..	..	..	..	..	29
11.0	<u>INSTALLATION LIMITS</u>	..	..	..	..	..	..	..	30
	<u>APPENDIX 1</u>	List of Compile Time Errors	..	..	..	..	..	..	31
	<u>APPENDIX 2</u>	List of Run Time Errors	..	..	..	..	..	..	38
	<u>APPENDIX 3</u>	ICL 4100 Card Code	..	..	..	..	..	..	41

PROGRAMMING IN ALGOL FOR THE ICL 4130

1.0 Introduction

It is assumed that the reader is familiar with the programming language Algol 60 as described in the publications

BALFOUR & Crook	:	Algol Programming Notes	:	Dept. of Computer Science, Heriot-Watt University
McCracken	:	A Guide to Algol Programming	:	John Wiley
Nicol	:	Elementary Programming and Algol	:	McGraw-Hill
Rutishauser	:	Description of Algol 60	:	Springer-Verlag

etc.. The purpose of this document is to outline the particular version of Algol which has been implemented on the ICL 4100 range of computers.

Under normal circumstances at this University all Algol programs will be run under the control of what is known as the DES BATCH operating system which has been designed to accept a mixture of programs in Algol, Fortran and Neat. This is a card-oriented, batch processing system in which a number of jobs are stacked in the card reader and processed in turn by the system. The stages of a job are controlled by control cards which are used to minimise operator intervention.

2.0 The Structure of an Algol Job Deck

A typical Algol job comprises the following cards:-

```
&JOB;<job number>;<user's name>;
&ALGOL;
&LIST;
```

↑
program cards

↓
&RUN;

↑
data cards

↓
&END;

Note that:-

(i) A control card is distinguished by an ampersand (&) punched in column 1.

(ii) The &JOB control card requires a job number which is assigned to an individual or class by the Department of Computer Science. The user's name must also be given. Typical examples of &JOB cards are

Example 1

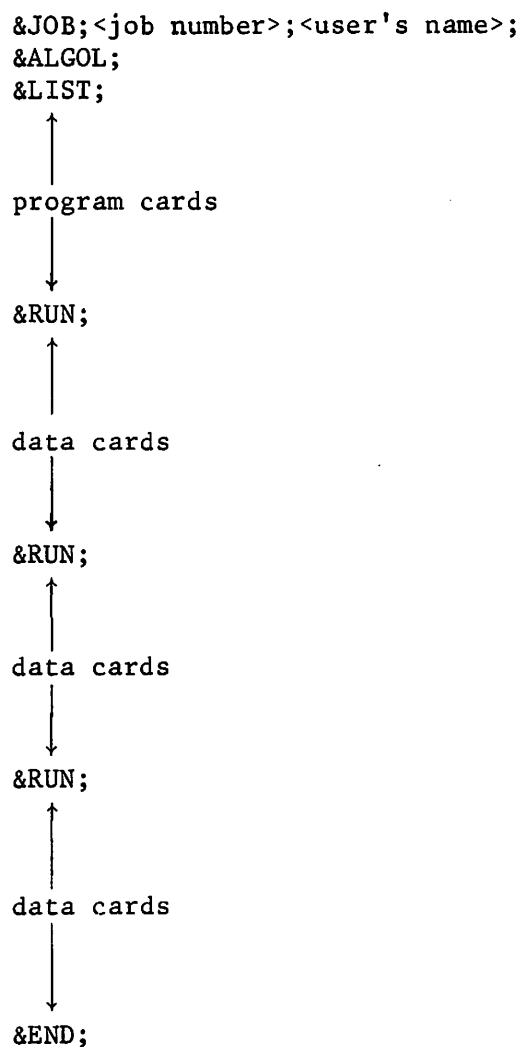
```
&JOB;999001;JOE BLOGGS;
```

Example 2

```
&JOB;999125;DAVID T CHALMERS;
```

(iii) The &LIST control card is optional. Its effect is to produce a listing of the program on the lineprinter.

(iv) The &RUN control card is used if the compiled program is to be executed by the computer. It is followed by the data (if any) for the run in question. The same program may be processed with several sets of data by using a number of &RUN cards thus:-



(v) The &END control card is the last card of a job.

3.0 The ICL 4100 Representation of Algol 60 Symbols

The IBM 029 card punches used for punching up program card decks do not possess all the symbols which occur in Algol 60. Some form of representation is therefore required for certain Algol symbols and the following has been adopted in ICL 4100 Algol:-

<u>ALGOL Symbol</u>	<u>ICL 4100 Representation</u>
A to Z	A to Z
a to z	A to Z
<u>begin</u> , <u>end</u> , <u>real</u> etc.	"BEGIN", "END", "REAL" etc.
+, -, /, ↑	+, -, /, ↑
x	*
÷	"DIV"
<, >, =	<, >, =
≤	"LE"
≥	"GE"
≠	"NE"
Λ, V	"AND", "OR"
⊃, ≡, ⊏	"IMP", "EQUIV", "NOT"
. , : ; :=	. , : ; :=
10	10
(,), [,]	(,), [,]
' , '	` , `
└	└

Note that:

- (i) "GO TO" or "GOTO" are both acceptable but not "GO" "TO".
- (ii) integer array is punched as "INTEGER" "ARRAY" not "INTEGER ARRAY".
- (iii) Special functions are punched in upper case letters e.g. SIN(X), EXP(X) etc.
- (iv) It is a matter of regret that when certain keys are depressed on our card punches the printing which appears at the top of the card does not agree with that on the key in question. To be more specific

\$	will print as	£
£	will print as	⌘
^	will print as	'
`	will print as	
↑	will print as	┐
[	will print as	\$
]	will print as	!
10	will print as	?

Do not be perturbed by this! When the program is listed on the lineprinter the correct characters will be printed.

4.0 ICL 4100 Input, Output Statements

The ICL 4100 input and output statements are easy to use but at the same time are extremely powerful and flexible. Note that with the DES BATCH operating system unless the programmer indicates to the contrary all numbers read in will be read from the card reader and all results printed out will be printed out on the lineprinter.

4.1 The Input of Information

A typical input statement in ICL 4100 Algol is

```
"READ" A,WILLY,J,X[I]
```

which has the effect of reading four numbers from cards and assigning their values to the variables with names A, WILLY, J and X[I]. As many variable names as required may be listed after the word "READ" provided they are separated from each other by commas.

Example 1

```
"READ" T,T5,X,GAMMA,TEMP,Y[5],K
```

Example 2

```
"FOR" J:=1"STEP"1"UNTIL"N"DO""READ"X[J]
```

Numbers to be read by the computer should, of course, conform to the Algol definition of a number. In particular:-

(i) Each number must be terminated by some character other than a digit, a decimal point, or a ₁₀. The use of a semi-colon or a comma is recommended for this purpose.

This terminal character may be followed by any sequence of characters, of any length, excluding the digits, a decimal point, a ₁₀, a ' , or a plus or minus sign. The only effect of the sequence of characters is to separate the numbers to be input and it is otherwise completely ignored.

(ii) Spaces must not be used in the middle of numbers since they are classified as terminating characters.

(iii) The Boolean values "true" and "false" may be read in and should be represented, in data, by the letters T and F respectively. When a Boolean variable is read all characters are ignored except and the digits until a letter T or a letter F is encountered. At all other times T and F retain the effects of normal text.

(iv) Columns 1 to 80 may be used for data information on any card.

Example 3 Given the Algol program:-

```
"BEGIN"
    "REAL X,Y,A7;
    "INTEGER" I,KAPPA;
    "BOOLEAN" TEST;
    "READ" X,A7,I,TEST,KAPPA;
    .
    .
    .
    "END";
```

and the data cards

36921;T;-614;

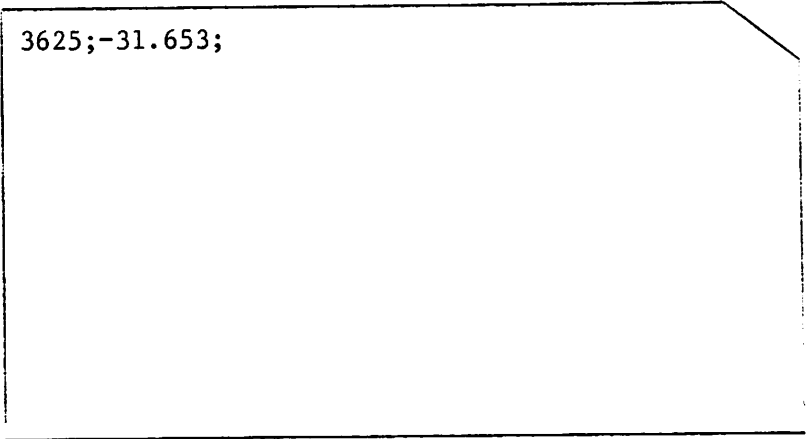
-2.635;418.923;

then the values -2.635, 418.923, 36921, true and -614 will be assigned to X, A7, I, TEST and KAPPA respectively.

Example 4 Given the Algol program:-

```
"BEGIN"
  "REAL" X,Y,T;
  "INTEGER" J,K;
  "READ" J;
  K:=J+1;
  "READ" X;
  .
  .
  .
  "END";
```

and the data card



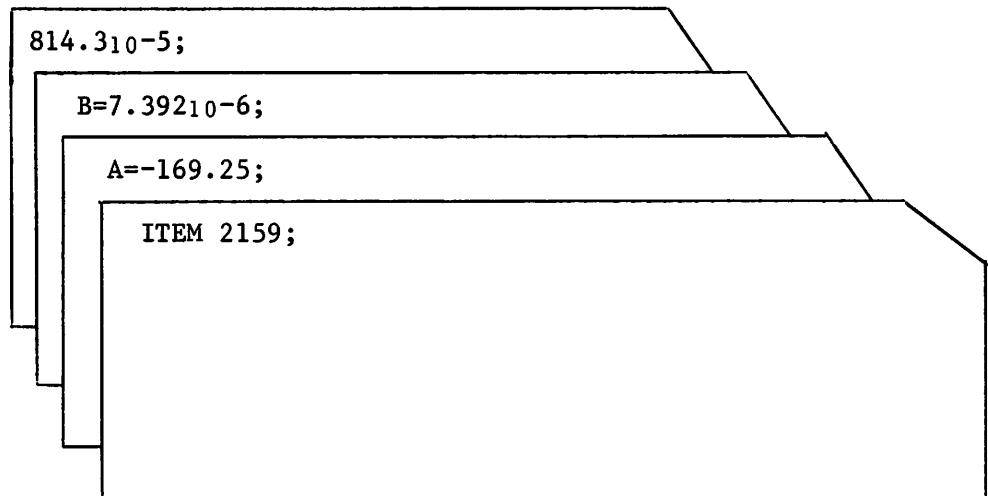
3625;-31.653;

then the first input statement "READ" J will assign the value 3625 to J and the second input statement "READ" X will assign the value -31.653 to X. Note that this is completely different from FORTRAN where each input statement triggers off the reading of a fresh data card.

Example 5 Given the Algol program:-

```
"BEGIN"
    "REAL" A,B,C;
    "INTEGER" ITEM;
    "READ" ITEM, A,B,C;
    :
    :
    :
"END";
```

and the data cards



then the values 2159, -169.25, 7.392×10^{-6} and 814.3×10^{-5} will be assigned to ITEM, A, B and C respectively.

4.2 The Output of Results

A typical output statement in ICL 4100 Algol is

```
"PRINT" T5,X12,ALPHA+BETA-7.6,COS(X+Y)
```

which has the effect of printing out, on the line-printer, the values of T5, X12, ALPHA+BETA-7.6 and COS(X+Y). As many variable names and/or arithmetic expressions may be listed after the word "PRINT" provided they are separated from each other by commas.

Example 1

```
"PRINT" A,B,C,(-B+SQRT(B*B-4.0*A*C))/(2.0*A)
```

Example 2

```
"FOR" I:=1"STEP"1"UNTIL"M"DO
"FOR" J:=1"STEP"1"UNTIL"N"DO""PRINT"A[I,J]
```

Note that, unless the programmer specifies to the contrary, all numbers are output under presumed settings. This means that

(a) Numbers printed by the computer will appear each at the start of a new line with up to eight digits printed.

(b) The values of integer quantities will be printed with the "sign floated" i.e. with leading zeros suppressed and, in the case of negative numbers, the first significant digit will be preceded by a minus sign.

(c) The values of real quantities, if less than 10^8 in absolute value, will be printed as decimal numbers containing eight digits, and preceded by a space if positive and by a minus sign if negative. Larger numbers will be printed in scaled format with four significant digits and followed by an exponent part.

Example 3 Examples of printing under presumed settings are:-

Integer Numbers

							9
				-	1	2	7
-	2	1	6	0	3	5	2
		5	1	3	2	9	0
	6	1	5	2	9	3	4

Real Numbers

-	9	.	0	0	0	0	0	0
	1	2	7	.	3	0	6	1
-	2	3	9	6	1	.	0	0
	.	0	0	0	0	0	0	0
-	1	.	6	5	2	₁₀	+	2

4.3 The Output of Text

To cause the computer to output headings and other messages, the characters which comprise the message are enclosed within the string quotes ` and ` , and included as an item in the list of an output statement. Note that the output of a text is not preceded by a change to a new line so that text will be printed on the same line as the last information output.

Example 1 The statement

```
"PRINT" X,`FEET`,Y,`INCHES`
```

might produce the output:-

1	3	.	6	1	2	3	5	2	F	E	E	T		
8	.	2	1	6	5	3	7	9	I	N	C	H	E	S

Example 2

```
"PRINT" Y,`IS THE VALUE OF THE FUNCTION`
```

might produce the output

```
-21.632895IS THE VALUE OF THE FUNCTION
```

To improve the lay-out of outputted information a number of special interpreted characters are available. Each interpreted character is surrounded by single string quotes ` and ` and then inserted into a piece of text (so that effectively each interpreted character is surrounded by double string quotes). The most useful of the interpreted characters are L and S which have the effects of triggering off a new line of output and leaving a space in output respectively.

One final point - it is permissible to follow an interpreted character by an unsigned integer e.g. 'L3', 'S6' etc.. This has the effect of repeating the action of the interpreted character the indicated number of times.

Example 3 The statement

```
"PRINT" X,Y,`L` END OF OUTPUT`
```

might produce the output

-	3	.	6	3	2	5	1	7	5								
	2	.	8	1	4	9	8	6	0								
E	N	D			O	F			O	U	T	P	U	T			

One further point with regard to the output of text is best explained by example. Suppose that we would like our lineprint output headed:-

```
NUMERICAL INTEGRATION BY REPEATED APPLICATION OF SIMPSONS RULE
```

Then we might be tempted to punch, say from column 8,

```
"PRINT" `L2`NUMERICAL INTEGRATION BY REPEATED APPLICATION OF SIM  
PSONS RULE`L2``;
```

where the M of SIMPSON has been punched in column 72. If we did so then our heading would appear as

```
NUMERICAL INTEGRATION BY REPEATED APPLICATION OF SIM  
PSONS RULE
```

- not quite what we wanted. This is because in the Elliott 4100 Algol system column 73 is interpreted as a "newline" character and in the above is output as such, just as if we had written 'L' in our own text.

One solution in the above case is to use two separate "PRINT" statements:-

If all the numbers to be printed on one line are to be preceded by the same character or characters use may be made of the procedure `PREFIX('-----')` which will cause the text displayed between `'` and ``` to be output (in the place of "new line") before every number printed. Note that if the procedure `PREFIX('-----')` is used then the procedure `SAMELINE` is unnecessary. In fact `SAMELINE` will cancel the effect of `PREFIX` if it occurs after it in the list, and vice-versa.

Example 3 The statement

```
"PRINT" X,PREFIX('***`'),B,C
```

might produce the output

```
-|2|. |1|1|2|3|6|7|4|*|*|*| |4|6|3|. |9|1|5|8|2|*|*|*|-|4|2|. |8|1|6|5|4|7|
```

Example 4 The statement

```
"PRINT" X,PREFIX(`;`S2```),Y,Z,A
```

will cause the values of X, Y, Z, A to be output on one line separated by a semicolon and two spaces.

4.4.2 Format Setting Procedures

(a) When Printing Values of Type Integer

Normally integers are output with up to eight digits printed. To specify the number of digits to be printed use the setting procedure `DIGITS(n)` where $1 \leq n \leq 8$.

Example 1 If I, J, K are of type integer the statement

"PRINT" I,DIGITS(4),J,K,L

might produce the output

		-	3	1	6	4	2	8
-	6	1	5	4				
			3	5				
		1	7	1				

where the value of I has been output under the control of the presumed setting (8 digits) and the values of J,K,L have been output under the control of DIGITS (4) i.e. with up to 4 digits printed.

Note that the procedure DIGITS(n) has no effect on the printing of real quantities.

(b) When Printing Values of Type Real

To achieve the same effect on real quantities as DIGITS(n) on integer quantities use the setting procedure FREEPOINT(n) where $1 \leq n \leq 12$.

Example 2 The statement

"PRINT" X,SAMELINE,FREEPOINT(5),Y,Z,A

might produce the output

-	6	1	4	3	.	1	2	5	7		1	2	.	9	8	4	-	5	1	6	.	2	4		.	0	0	0	0	8
---	---	---	---	---	---	---	---	---	---	--	---	---	---	---	---	---	---	---	---	---	---	---	---	--	---	---	---	---	---	---

A number of other format setting procedures are available when outputting quantities of type real. For example to output a real value with up to m digits before the decimal point and n digits after it use the setting procedure ALIGNED(m,n) where $m+n \leq 15$

Example 3 The statement

```
"PRINT" ALIGNED(4,3),X,Y,Z,ALPHA
```

might produce the output

3	1	7	6	.	2	1	0
-	8	3	.	0	1	7	
-	1	2	6	5	.	8	2
			0	.	1	7	7

Another useful format setting procedure for outputting real quantities is the procedure `SCALED(n)` where $1 \leq n \leq 12$. This causes values to be output in the form

$$X.XXX-----X_{10}^{\pm XX}$$

where there are n digits before the exponent part and the decimal point is positioned after the first digit.

Example 4 The statement

```
"PRINT" SCALED(6),X,Y,Z,ALPHA
```

might produce the output

-	2	.	5	2	7	3	4	₁₀	+	0	7
	3	.	2	1	8	9	0	₁₀	-	1	4
	3	.	8	2	6	5	4	₁₀	-	0	2
-	8	.	1	1	1	2	2	₁₀	+	1	3

Example 5 The statement

```
"PRINT" FREEPOINT(4),X,PREFIX('`,`S2`'),SCALED(4),Y,Z,ALIGNED(5,4),ALPHA
```

might produce the output

-	2	1	3	.	4	,				1	.	3	7	6	₁₀	+	0	4	,							3	6	5	.	7	3	1	8
---	---	---	---	---	---	---	--	--	--	---	---	---	---	---	---------------	---	---	---	---	--	--	--	--	--	--	---	---	---	---	---	---	---	---

Note that the format setting procedures `FREEPOINT(n)`, `ALIGNED(m,n)` and `SCALED(n)` have no effect on the printing of integer quantities.

(c) When Printing Values of Type Boolean

The printing of Boolean values is unaffected by DIGITS(n), FREEPOINT(n), ALIGNED(m,n), SCALED(n), SAMELINE and PREFIX(`-----`).

Example 6 If A, B and C are of type Boolean then the statement

```
"PRINT" `L`,A,`S2`,B,`S2`,C
```

might produce the output

```
|T| | |F| | |F|
```

4.4.3 Device Setting Procedures

In order to use input and output devices other than the card reader and the line-printer the device setting procedures READER(n) and PUNCH(n) must be inserted at the appropriate place in a program. The value to be assigned to n in a particular situation can be taken from the following tables:-

READER (n)

n = 1	paper-tape reader 1
n = 2	paper-tape reader 2*
n = 3	on-line teleprinter
n = 6	card reader

PUNCH (n)

n = 1	paper-tape punch 1
n = 2	paper-tape punch 2*
n = 3	on-line teleprinter
n = 4	lineprinter
n = 5	digital plotter
n = 6	card punch*

* These devices are not available on the Heriot-Watt 4130 configuration.

Example 1 The statement

```
"PRINT" X,Y,Z,PUNCH(1),X,Y,Z
```

will output the values of X, Y, Z on the lineprinter and also on punched paper-tape.

Example 2 The statement

```
"READ" READER(1),A,B,ALPHA
```

will input three numbers from paper-tape reader 1 and assign their values to A, B and ALPHA.

Example 3 The statement

```
"PRINT" ^L`SUM=`,SUM,PUNCH(1),X,Y,Z,  
PUNCH(3),^L`FEED NEXT TAPE
```

will

- (i) output the message SUM= on the lineprinter and follow it by the value of SUM.
- (ii) output the values of X,Y,Z on paper-tape.
- (iii) output the message FEED NEXT TAPE on the control teleprinter.

4.4.4 Parameters out of Range

If the parameter(s) of a setting procedure are outside the permitted range, a return to the presumed settings will occur as follows:-

<u>Setting Procedure</u>	<u>Permitted Range of Parameters</u>	<u>Presumed Setting</u>
DIGITS(n)	$1 \leq n \leq 8$	DIGITS(8)
FREEPOINT(n)	$1 \leq n \leq 12$	FREEPOINT(8)
ALIGNED(m,n)	$1 \leq m+n \leq 15$	FREEPOINT(8)
SCALED(n)	$1 \leq n \leq 12$	FREEPOINT(8)
READER(n)	$1 \leq n \leq 6$	READER(6)
PUNCH(n)	$1 \leq n \leq 6$	PUNCH(4)

4.4.5 Alarm Printing

If an attempt is made to print a number which is too large for the style of printing specified (e.g. to print 289.2 using ALIGNED(2,5) or to print 12345 using DIGITS(4)) then "alarm" printing will occur in such a way that the layout of the printed page will be undisturbed. Provided that the total number of characters called for is greater than 6, the number will be printed in scaled format so that it occupies the same space as the programmer allowed. On the other hand if the number of characters called for is less than 6 (making scaled format impossible) then a capital H is printed in the space allowed by the programmer for the least significant digit of the number.

5.0 A Sample Job for the DES BATCH Operating System

We consider the following simple problem:-

Values for the sides b,c and the angle A of $\triangle ABC$ are given on a card. Calculate and print the length of the remaining side and print also the length of the shortest side of the triangle. Assume that the angle A is given in degrees.

A possible program is given on the attached sheet. Note that:-

- (i) Control cards are punched from column 1.
- (ii) A title for the program must be given immediately before the first "BEGIN" of the Algol program. This title must consist of capital letters and digits only and must be terminated by a semi-colon. The initial character must be a letter. In our sample job the title is, of course, TRIGCALC.

NAME		A BALFOUR		PROGRAM TITLE		TRIGCALC		DATE		4/2/72											
ADDRESS DEPT. OF COMPUTER SCIENCE				User No.		129001		Language		Algol											
				Data Format				Page		1 of 1											
Sequence No.																					
1	2	5	6	7	10	15	20	25	30	35	40	45	50	55	60	65	70	73	75	80	
250001, 129001, A BALFOUR;																					
BALGOL;																					
TRIGCALC;																					
BEGIN																					
REAL A, B, C, ANGA, MINSIDE;																					
READ B, C, ANGA;																					
PRINT "2, 1, TRIG CALCULATION PROGRAM A BALFOUR 4/2/72";																					
A:=SQRT(B*B+C*C-2.0*B*C*Cos(3.1415926*ANGA/180.0));																					
MINSIDE:=A;																					
IF B<MINSIDE THEN MINSIDE:=B;																					
IF C<MINSIDE THEN MINSIDE:=C;																					
PRINT "SIDE A = , SAMELINE, SCALED(6), A";																					
PRINT "1, 1, SHORTEST SIDE = , SAMELINE, SCALED(6), MINSIDE																					
END																					
RUN																					
1.73 36.94 73.0																					
END;																					

(iii) The final 'END' in an Algol program must be terminated by a semi-colon.

(iv) Each line of an Algol program is written in free format between columns 1 and 72 inclusive. If a line is too long to be punched on one card, in this way, it should be written on successive lines between columns 1 and 72. No indication is required on the continuation line.

(v) The last two PRINT statements could have been written

```
SAMELINE ;
SCALED(6);
"PRINT" 'SIDE A = `, A;
"PRINT" 'L`SHORTEST SIDE = `,MINSIDE
```

or alternatively

```
SAMELINE;
SCALED(6);
"PRINT" 'SIDE A = `,A,'L`SHORTEST SIDE = `,MINSIDE
```

(vi) Great care should be taken to indicate clearly which character is required. Confusion can arise between:-

Letter I (write I)	and figure 1 (write 1)
Letter O (write ☐ or $\emptyset$)	and figure 0 (write 0)
Letter Z	and figure 2
Letter S	and figure 5
Letter U	and letter V
Letter H	and letter M
Letter E	and letter G
Letter C	and bracket (
Letter D	and letter O
Figure 1	and figure 7
Letter G	and figure 6

This is particularly important when the coding sheets are to punched up by some other person.

(vii) To assist the operators running programs under the DES BATCH operating system the &JOB card should be orange in colour and all other control cards should be green in colour. Plain or yellow cards are also recommended for program and data information. Adherence to this system of colours enables the operators to separate jobs easily by using the single orange card as a guide to the start of a new job.

(viii) A number of short instructions may be punched on the one card provided that, of course, they are separated by semi-colons e.g.

$$T:=X;X:=2*Y;Y:=T;$$

6.0 Restrictions on Algol 60

In 4100 Algol a number of restrictions are imposed on the full generality of Algol. The more important restrictions are as follows:-

(i) Only the first six characters of an identifier are of consequence e.g. the identifiers CONDENSER and CONDENSATION are regarded by the system as being identical.

(ii) The result of the expression $P \uparrow Q$ is always real, irrespective of the type of P or Q. This means that, for example, the statement

$$I := N * (N + 1) \uparrow (2 \uparrow 3)$$

where I, N are integers, is illegal (why?).

(iii) Unsigned integers must not be used as labels.

(iv) All formal parameters of a procedure must be specified in the head of the procedure.

(v) The lower bound of an array subscript must be in the range -256 to 255.

(vi) The values of simple subscripts of subscripted variables on the left-hand-side of assignment statements are not preserved. For example the statements

$$\begin{array}{l} I:=5; \\ A[I]:=I:=I+1 \end{array} \quad \leftarrow \text{assignments go } R \rightarrow L$$

would assign the value 6 to A[6] and not to A[5]. This is because the I of A[I] is a simple subscript. The normal Algol effect can be achieved by writing

$$A[I+0] := I := I+1 \quad \leftarrow (\text{assignments in direction } R \rightarrow L)$$

There are a number of other restrictions on the full generality of Algol 60 mainly concerned with procedures and their parameters. The interested reader should refer to pages 7 to 10 of Vol. 2, Part 2, Section 2 of the ICL 4100 Programming Information.

7.0 Additions to Algol 60

These are described in pages 33 to 46 of Vol. 2, Part 2, Section 2 of the ICL 4100 Programming Information. Of interest is the fact that as well as the standard Algol Functions

ABS(X)	EXP(X)	LN(X)	SQRT(X)
SIN(X)	COS(X)	ARCTAN(X)	
SIGN(X)	ENTIER(X)		

the additional functions

TAN(X)	ARCSIN(X)	ARCCOS(X)
--------	-----------	-----------

are also provided. In all cases no explicit declaration is required in a program and X may be an arithmetic expression.

8.0 Range and Accuracy of Numbers

(a) Integer Quantities

The value of a quantity of type integer must always lie between -8388608 (i.e. -2^{23}) and 8388607 (i.e. $2^{23} - 1$) inclusive.

(b) Real Quantities

The value of a quantity of type real must lie between the approximate limits -5.8×10^{76} and 5.8×10^{76} . All real quantities within this range are held to an accuracy of about 11 significant decimal digits with the one exception that quantities less than 4.3×10^{-76} in magnitude cannot be distinguished from zero.

9.0 Hints on Efficiency

A knowledge of the following points will enable a programmer to save either computer time or computer storage space (or both) without causing great inconvenience to the programmer concerned. Do not however become so obsessed with efficiency that the clarity and simplicity of your Algol programs are impaired!

(i) Wherever possible avoid the mixing of real and integer quantities in the one instruction. For example

$$X := (-B + \text{SQRT}(B*B - 4.0*A*C)) / (2.0*A)$$

is very much better than

$$X := (-B + \text{SQRT}(B*B - 4*A*C)) / (2*A)$$

(ii) If arrays (of the same type) have identical subscript bounds, it will save space if they are declared as a single array segment. By way of explanation

"ARRAY" A,B,C [1:10, 1:20]

is very much better than

"ARRAY" A[1:10,1:20],B[1:10,1:20],C[1:10,1:20]

(iii) In array declarations space will also be saved if the subscripts are in increasing order of range from left to right. By way of explanation

"ARRAY" A[1:2, 1:100]

is very much better than

"ARRAY" A[1:100, 1:2]

(iv) If I, J are of type integer then

(a) I+I is 2 times faster than 2*I

(b) J:=I*I is 4 times faster than J:=I↑2

(c) J:="IF" I<0 "THEN" -I "ELSE" I is 25 times faster
than J:=ABS(I)

(v) In for statements of the form

"FOR" X:=A "STEP" B "UNTIL" C "DO" - - - - -

time will be saved by making B,C simple variables, wherever possible. If B,C are expressions then they will be evaluated anew for each value of the controlled variable X.

(vi) Space can be saved by not using the logical operators "AND", "OR", "NOT", "IMP" and "EQUIV" wherever possible. For example

"IF" I>0 "THEN" "BEGIN" "IF" J>2 "THEN" - - - - -

is very much better than

"IF" I>0 "AND" J>2 "THEN" - - - - -

(vii) A switch declaration should only be made if it is essential.

By way of an example a switch list of 5 simple labels will occupy 52 locations in total.

(viii) Expressions involving polynomials are evaluated much more efficiently when written in "nested" form. For example

$$Y := ((A * X + B) * X + C) * X + D$$

is very much better than

$$Y := A * X^3 + B * X^2 + C * X + D$$

(ix) Do not call array parameters by value without good reason - this will save both space and time. Even more important call all other parameters by value unless it is absolutely essential to call them by name. Even when it is known that the actual parameter replacing a formal parameter is a simple variable the formal parameter should be called by value. The failure to call parameters by value, wherever possible, is the greatest single cause of inefficiency in Algol programs.

(x) Where the effect of a parameter specified as a procedure can be achieved equally well by a parameter called by name, it will usually be quicker and result in less object code to use the name parameter.

(xi) The controlled statement of a for statement is often changed from a compound statement to a block by the presence of declarations within it. A non-negligible amount of computer time can be saved by avoiding this wherever possible.

10.0 The Algol Error Diagnosis System

Error messages are output by the system during the processing of an Algol job. Two types of error message may be output:-

(a) compile time errors i.e. errors in syntax detected by the system during the compilation of the object program from the source program.

(b) run time errors i.e. errors which are detected during the running of the object program.

10.1 Compile Time Errors

If a syntax error is detected during compilation the message

ALG ERR <unsigned integer>

is output to the lineprinter, starting at column 84. If the offending symbol is an identifier then the above message is followed by the identifier in question.

Example 1 ALG ERR 40

Example 2 ALG ERR 32 BETA

A list of compile time errors and their associated identification numbers is given in Appendix 1 to this document. For example

ALG ERR 40

means that there is a "THEN" missing in an if statement and

ALG ERR 32 BETA

means that the identifier BETA has either not been declared or has been used outside the scope of its declaration.

Note that when a number of compile time errors are output one or more of these errors may be "spurious". This occurs when a syntax error early in a program "confuses" the compiler causing it to misinterpret correct Algol syntax which appears later.

10.2 Run Time Errors

Other errors may occur during the course of a program run. For example if, in a program, we try to evaluate the square root of a negative number then the system will output on the lineprinter the message

SQRT ERR

and if we try to evaluate the inverse sine of a quantity which is greater than unity in magnitude then the system will output the message

ARCS ERR

A list of run time errors is given in Appendix 2 to this document.

Note that in certain cases the system will continue with a program after outputting an error message. Thus, for example, if we have in a program

Y:= SQRT(X)

and $X < 0$ when this instruction is obeyed then the system will output the message

SQRT ERR

and continue with Y set equal to zero.

10.3 State of a Job on Completion

At the end of a job, after the time taken has been output, a single character is output to the lineprinter (and to the teletype) to show the state of the job on completion. The meaning of the character output is as follows:-

- A The job ran to completion.
- B The run was terminated by an &RUN, &END or &JOB card being read in the data.
- C A control card was encountered in the wrong position, or it was not possible to perform the job as specified by the control cards.
- E The operator abandoned the job, or a non-continuable software error occurred.
- F The program failed to compile.
- G The program did not run correctly.
- H The programmer's specified time expired.
- I The programmer's specified number of lines was exceeded.
- J An incorrect control card was encountered.
- U The expected time exceeded the installation maximum time.
- V The installation default time expired.
- W The installation maximum time expired.
- X The installation maximum number of lines was exceeded.
- Z A hardware failure occurred (e.g. a read failure on magnetic tape).

11.0 Installation Limits

Each 4100 installation is able to control

- (i) the maximum time a given job may run
- (ii) the maximum number of lines of output a given job may produce.

At Heriot-Watt these installation limits are set at 1 minute and 1200 lines respectively.

Any user wishing to exceed these limits must submit his job in a job bag indicating his specific requirements. Failure to do so will result in his job being terminated if either limit above is attained.

APPENDIX 1COMPILE TIME MESSAGESNo.

- 1 Number of impermissible form.
- 2 Error in basic word.
- 3 Impermissible beginning to a statement.
- 4 Procedure declaration not terminated by a semicolon.
- 5 Name in declaration not terminated by semi-colon or comma.
- 6 Names in call of "LIBRARY" not present in Library. (This error is followed by a list of the names not found).
- 7 Name declared twice in same blockhead.
- 8 Label occurring twice in same block.
- 9 Item in a declaration inadmissible.
- 10 First item in switch declaration not followed by :=

In a procedure declaration (11 to 21):

- 11 Item following a procedure name not semicolon or open bracket.
- 12 No semicolon or close bracket after formal parameter part.
N.B. If the item following the close bracket is not a semicolon the compiler will ignore a letter string until :.
- 13 List in value or specification part has impermissible form.
- 14 Specification part occurs before value part.
- 15 More than 10 (ten) parameters are specified as procedures.
- 16 Too many parameters (more than 24).
- 17 Parameter not specified.
- 18 Recursive procedure encountered with real, integer, label or Boolean name parameter.
- 19 Name in value part not a formal parameter.

32

- 20 Name in specification part not a formal parameter.
- 21 Parameter specified twice.
- 22 Program name of impermissible form. (After this error, the program name is changed to ANON and compilation continues normally).

In array declarations (23 to 26):

- 23 No comma or open bracket after identifier.
- 24 No colon between upper and lower bounds.
- 25 No comma or closed bracket after bound pair.
- 26 Array with too many dimensions (more than 63).
- 27 No "END" or semicolon after statement in compound tail.
- 28 No colon after a label.
- 29 Number of words required for variables or for constants exceed 4095.
- 30 Left part variable in assignment statement not followed by :=
- 31 Value assigned to a procedure identifier outside procedure body, or type procedure name used within a NEAT segment.
- 32 Identifier not declared, or used outside scope of declaration.
- 33 Inadmissible complex primary in arithmetic expression.
- 34 Missing arithmetic operand at start of arithmetic expression.
- 35 Missing operand in arithmetic expression.
- 36 No close bracket after subscript list.
- 37 Unmatched brackets.
- 38 Wrong number of subscripts in subscripted variable.
- 39 Missing "ELSE".
- 40 Missing "THEN".
- 41 Conditional statement or expression after "THEN".
- 42 Missing or inadmissible operator in arithmetic expression.
- 43 Non arithmetic operand in arithmetic expression.

- 44 Impermissible use of label name.
- 46 Inadmissible identifier as controlled variable in "FOR" statement.
- 47 Missing operand at start of Boolean expression.
- 48 Missing relational operator.
- 49 Missing operand in Boolean expression.
- 50 Inadmissible complex Boolean primary.
- 51 Inadmissible operator in Boolean expression.
- 52 Inadmissible symbol at start of an expression.

During procedure call (53 to 56):

- 53 No open bracket following name of procedure with parameters.
- 54 Actual parameter not followed by comma or close bracket (cf. error 12).
- 55 Error in parameter delimiter of form)\letter string>:(
- 56 No actual switch procedure or string parameter, where one expected.
- 58 Controlled variable in "FOR" statement not followed by :=
- 60 Incorrect designational expression.
- 61 Arithmetic expression in "FOR" list element not followed by "STEP"
"WHILE" "DO" or a comma.
- 62 Missing "UNTIL".
- 63 Program too large or complex to be compiled (e.g. too many
declared identifiers).
- 64 Occurrence of a ' within inner string.
- 65 "COMMENT" occurs otherwise than after a semicolon or a "BEGIN".
- 66 Conditional arithmetic expression on right-hand side of relation.
- 67 "GO TO" into a "FOR" loop from outside.
- 68 Amount of code produced for current ALGOL block exceeds 16,383 words.
- 69 Wrong number of parameters in procedure call.
- 70 Formal array parameter not replaced by array name.
- 71 Name output parameter replaced by a variable of a different type,
a complex expression, or a constant.

- 72 Error in segment of NEAT code.
- 73 Jump to one or more non-existent labels in NEAT code segment.
- 74 Error in switch declaration.
- 75 Own array with variable bounds.
- 76 Symbol following "OWN" not "REAL", "INTEGER" or "BOOLEAN".
- 77 (i) In the call of a formal procedure:
 The type and class of an actual parameter do not agree with
 that expected.
- (ii) In the call of a procedure having parameters which are
 specified as procedures:
 An actual procedure possesses parameters which are not of
 the same type or class as those of the corresponding formal
 procedure.

In the call of a procedure having parameters which are specified as
procedures (78 to 80):

- 78 An actual procedure possesses name output parameters, and these
 do not correspond to simple variables in all the calls of the
 corresponding formal procedure (cf. error number 71).
- 79 An actual procedure does not have the same number of parameters
 as the corresponding formal procedure.
- 80 An actual procedure is not of the same type as the corresponding
 formal procedure, or is a function designator when the corresponding
 formal procedure is not, or vice-versa.

Any number greater than 1000:

This indicates that the compiler cannot continue translation. It
usually occurs after a number of other errors.

Other Errors during Translation

- (1) The display of the message:

ERCHAR1

can only occur when compiling from paper tape and indicates either:

- (i) An odd parity character has been read.
- or (ii) A character not acceptable to 4100 ALGOL has been read. This includes all those characters which are not in the 4100 extended character set, and $\leftarrow$ and !

The wrong character will be the last character input by the tape reader. To continue compilation, ignoring the wrong character, type . (full stop). If any other character(s) are typed, the compilation is abandoned.

- (2) The message

CARE

is displayed if the comment between "END" and the following "END" "ELSE" or semi-colon does not have the form of a single legal ALGOL identifier. The message will never be output more than once for any "END" in the program. It is only a warning and compilation continues normally.

- (3) If one or more references to non-existent labels or to labels outside the scope of their blocks occur then the message

NOLABEL

is displayed followed by a list of the offending label names, printed in upper case. If any of the names are in lower case, then an * is printed before each such name. It is possible to run the program, but if any of the labels are used, the program is terminated.

- (4) If the program is punched on a paper tape, the absence of a halt code at the end of the tape will cause it to shoot through the reader, instead of stopping at the end.
- (5) The following errors may cause the ALGOL program to be completely read, without finishing compilation (for example on the B20 or T20 ALGOL systems, the message RELOAD is displayed):
 - (i) Insufficient "END"s to match all the "BEGIN"s in the program.
 - (ii) No semi-colon after the final "END".
 - (iii) Missing ` at the end of a string, causing the program statements that follow to be treated as part of the string. (This error will be detected (ERROR 64) if an inner string occurs in any subsequent statement).
- (6) The following errors will cause the end of an ALGOL program to be found prematurely:
 - (i) Missing "BEGIN".
 - (ii) "END" or the comment following "END" not followed by "END", "ELSE" or semi-colon, causing a "BEGIN" to be treated as part of a comment.

These errors lead to breakdown of the block structure of ALGOL and will usually cause spurious error messages to be displayed.

- (7) If an editing error occurs, the message output and the action taken should be the same as for EDIT 41 (see Section 2.14.2 of the 4100 Technical Manual).
- (8) The message

PTPILOW

indicated that more tape is required. After reloading the tape punch, continuation is effected by typing a full stop.

The resulting IN-code tapes are quite suitable for input, provided the last one is input first, and the first one is input last.

- (9) When compiling to store, the message

MCNOTACC

means that the main chapter of the compiled program is placed so that it is not accessible by direct orders. This will happen only on stores greater than 32K and if there are a number of other programs in the store. To rectify the condition, cancel one or more of the other programs and recompile.

- (10) The message

CANTLIST

indicates that no lineprinter is attached, when optional listing is required. The run will be abandoned.

- (11) The messages

and
NO INPUT MODULE
NO OUTPUT MODULE

indicate that either the input module or the output module is absent from store. The missing module should be loaded and the entry parameters retyped.

- (12) The messages

and
OPNOTSET
NOENTRY

indicate that the compiler and modules have been corrupted, and the whole system should be re-input.

RUN TIME MESSAGES

Message	Description	Continuation
READERR	(i) decimal point or exponent separator read in an integer; (ii) two decimal points read in a number; (iii) two exponent separators read in a number; (iv) decimal point read following an exponent separator; (v) no digits, or sign read following an exponent separator; (vi) no digits, decimal point or exponent separator read after a sign; (vii) no digits read following a decimal point; (viii) numeric character found before the first ` when obeying INSTRING;</li> <li>(ix) occurrence of a ` within an inner string when obeying INSTRING. (x) ` or digit read when reading Booleans.	- Continues with zero as input value. - Continues with zero as input value. - Continues with zero as input value. - Continues with zero as input zero. - Continues with zero as input value. - Continues with zero as input value. - Continues with zero as input value. - String terminated in store and next instruction obeyed. - String terminated in store and next instruction obeyed. - Continues with "FALSE" as input value.
INTOFLO	The result of some integer operation (e.g. division by zero) is outside the range -2^{23} to $2^{23}-1$. Note that the integer overflow is only detected on an integer multiplication or on "DIV", even though it may have been caused by some previous operation (for example, in a NEAT segment).	No continuation.
SWITOFLO	The value of the subscript in a switch designator is outside the range of the switch list.	The program immediately proceeds with the next statement, except in the case where the switch designator is used as an actual parameter, when SUBOFLO is displayed at the time this parameter is used.
SUBOFLO	(i) the subscript of a subscripted variable is outside the declared range; (ii) a switch designator used as an actual parameter: see SWITOFLO above.	- No continuation - No continuation
NOROOM	The program requires more space than is available in the computer.	No continuation

Messages	Description	Continuation
SQRT ERR	$X < 0$	Continues with SQRT set to zero.
SIN ERR	$ X > 8.5 \times 10^{11}$ (approximately)	Continues with SIN set to zero.
COS ERR	$ X > 8.5 \times 10^{11}$ (approximately)	Continues with COS set to zero.
TAN ERR	$ X > 4.2 \times 10^{11}$ (approximately)	Continues with TAN set to the largest positive number.
LOG ERR	$X < 0$ or $p \uparrow q$ with $p < 0$ and q real.	Continues with LOG set to zero.
EXP ERR	$X > 254 \log_e 2$ (i.e. 176.0)	Continues with EXP set to the largest positive number.
ARCS ERR	$ X > 1$	Continues with ARCSIN set to zero.
ARCC ERR	$ X > 1$	Continues with ARCCOS set to zero.
FPOFLO	Floating point overflow.	Continues with largest number of correct sign as the value used.
OUTSERR	The string being output by PREFIX or LEAD-ZERO or OUTSTRING contains an inadmissible character.	Ignores the rest of the string and continues.
PRNTERR	Non-zero real number to be output has a mantissa less in absolute value than $\frac{1}{4}$; the most likely cause of this is that the programmer has failed to assign a value to a variable before printing it.	Outputs zero and continues.
BUFF ERR	In a call of BUFFER (n, 's`):  (i) More than one character specified in the string. e.g. BUFFER (1, 'AB`'); (ii) No character in the string e.g. BUFFER (1, ``);  (iii) Inner string used. e.g. BUFFER (1, ``L``);	Continues with the value "FALSE".

Message	Description	Continuation
ENTOFLO	Integer overflow caused during a call of ENTIER.	No continuation
LOWBDERR	In a call of LOWBOUND, the second parameter is greater than the number of subscripts of the array.	No continuation.
RANGERR	In a call of RANGE, the second parameter is greater than the number of subscripts of the array.	No continuation.
BOUNDERR	An attempt has been made to allocate an array in which (i) One or more rows contains more than 32,765 integer or 16,382 real numbers. (ii) the lower bound is greater than the upper bound. (iii) the lower bound is outside the range -256 to +255.	No continuation.

APPENDIX 3ICL 4100 CARD CODE

The following card code is used in the DES BATCH operating system. The rows are named 12, 11, 0, 1, -----, 9 starting from the top of the card.

Character	Code	Character	Code
space	null	@	4-8
"	7-8	A	12-1
½	0-2-8	B	12-2
\$	11-3-8	C	12-3
%	0-4-8	D	12-4
&	12	E	12-5
' (acute)	5-8	F	12-6
(	12-5-8	G	12-7
)	11-5-8	H	12-8
*	11-4-8	I	12-9
+	12-6-8	J	11-1
, (comma)	0-3-8	K	11-2
-	11	L	11-3
.	12-3-8	M	11-4
/	0-1	N	11-5
0	0	O	11-6
1	1	P	11-7
2	2	Q	11-8
3	3	R	11-9
4	4	S	0-2
5	5	T	0-3
6	6	U	0-4
7	7	V	0-5
8	8	W	0-6
9	9	X	0-7
:	2-8	Y	0-8
;	11-6-8	Z	0-9
<	12-4-8	[	12-2-8
=	6-8	£	3-8
>	0-6-8	]	11-2-8
10	0-7-8	†	11-7-8
		` (grave)	12-7-8

